
Subject: [PATCH 01/12] proc: switch /proc/driver/ray_cs/ray_cs to seq_file interface
Posted by [Alexey Dobriyan](#) on Tue, 08 Apr 2008 14:50:49 GMT
[View Forum Message](#) <> [Reply to Message](#)

Signed-off-by: Alexey Dobriyan <adobriyan@sw.ru>

drivers/net/wireless/ray_cs.c | 73 ++++++-----
1 file changed, 42 insertions(+), 31 deletions(-)

--- a/drivers/net/wireless/ray_cs.c

+++ b/drivers/net/wireless/ray_cs.c

@ @ -34,6 +34,7 @ @

#include <linux/kernel.h>

#include <linux/proc_fs.h>

#include <linux/ptrace.h>

+#include <linux/seq_file.h>

#include <linux/slab.h>

#include <linux/string.h>

#include <linux/timer.h>

@ @ -2582,7 +2583,7 @ @ static char *nettype[] = {"Adhoc", "Infra "};

static char *framing[] = {"Encapsulation", "Translation"}

;

/*=====

==*/

-static int ray_cs_proc_read(char *buf, char **start, off_t offset, int len)

+static int ray_cs_proc_show(struct seq_file *m, void *v)

{

/* Print current values which are not available via other means

* eg ifconfig

@ @ -2606,83 +2607,93 @ @ static int ray_cs_proc_read(char *buf, char **start, off_t offset, int len)

if (!local)

return 0;

- len = 0;

-

- len += sprintf(buf + len, "Raylink Wireless LAN driver status\n");

- len += sprintf(buf + len, "%s\n", rcsid);

+ seq_puts(m, "Raylink Wireless LAN driver status\n");

+ seq_printf(m, "%s\n", rcsid);

/* build 4 does not report version, and field is 0x55 after memtest */

- len += sprintf(buf + len, "Firmware version = ");

+ seq_puts(m, "Firmware version = ");

if (local->fw_ver == 0x55)

- len += sprintf(buf + len, "4 - Use dump_cis for more details\n");

+ seq_puts(m, "4 - Use dump_cis for more details\n");

```

else
-   len += sprintf(buf + len, "%2d.%02d.%02d\n",
+   seq_printf(m, "%2d.%02d.%02d\n",
               local->fw_ver, local->fw_bld, local->fw_var);

for (i=0; i<32; i++) c[i] = local->sparm.b5.a_current_ess_id[i];
c[32] = 0;
-   len += sprintf(buf + len, "%s network ESSID = \"%s\"\n",
+   seq_printf(m, "%s network ESSID = \"%s\"\n",
               nettype[local->sparm.b5.a_network_type], c);

p = local->bss_id;
-   len += sprintf(buf + len, "BSSID          = %s\n",
+   seq_printf(m, "BSSID          = %s\n",
               print_mac(mac, p));

-   len += sprintf(buf + len, "Country code    = %d\n",
+   seq_printf(m, "Country code    = %d\n",
               local->sparm.b5.a_curr_country_code);

i = local->card_status;
if (i < 0) i = 10;
if (i > 16) i = 10;
-   len += sprintf(buf + len, "Card status      = %s\n", card_status[i]);
+   seq_printf(m, "Card status      = %s\n", card_status[i]);

-   len += sprintf(buf + len, "Framing mode     = %s\n", framing[translate]);
+   seq_printf(m, "Framing mode     = %s\n", framing[translate]);

-   len += sprintf(buf + len, "Last pkt signal lvl = %d\n", local->last_rsl);
+   seq_printf(m, "Last pkt signal lvl = %d\n", local->last_rsl);

if (local->beacon_rxed) {
/* Pull some fields out of last beacon received */
-   len += sprintf(buf + len, "Beacon Interval    = %d Kus\n",
+   seq_printf(m, "Beacon Interval    = %d Kus\n",
               local->last_bcn.beacon_intvl[0]
               + 256 * local->last_bcn.beacon_intvl[1]);

p = local->last_bcn.elements;
if (p[0] == C_ESSID_ELEMENT_ID) p += p[1] + 2;
else {
-   len += sprintf(buf + len, "Parse beacon failed at essid element id = %d\n", p[0]);
-   return len;
+   seq_printf(m, "Parse beacon failed at essid element id = %d\n", p[0]);
+   return 0;
}
}

```

```

    if (p[0] == C_SUPPORTED_RATES_ELEMENT_ID) {
-       len += sprintf(buf + len, "Supported rate codes = ");
+       seq_puts(m, "Supported rate codes = ");
        for (i=2; i<p[1] + 2; i++)
-           len += sprintf(buf + len, "0x%02x ", p[i]);
-       len += sprintf(buf + len, "\n");
+       seq_printf(m, "0x%02x ", p[i]);
+       seq_putc(m, '\n');
        p += p[1] + 2;
    }
    else {
-       len += sprintf(buf + len, "Parse beacon failed at rates element\n");
-       return len;
+       seq_puts(m, "Parse beacon failed at rates element\n");
+       return 0;
    }

    if (p[0] == C_FH_PARAM_SET_ELEMENT_ID) {
        pfh = (struct freq_hop_element *)p;
-       len += sprintf(buf + len, "Hop dwell          = %d Kus\n",
+       seq_printf(m, "Hop dwell          = %d Kus\n",
        pfh->dwel_time[0] + 256 * pfh->dwel_time[1]);
-       len += sprintf(buf + len, "Hop set           = %d \n", pfh->hop_set);
-       len += sprintf(buf + len, "Hop pattern       = %d \n", pfh->hop_pattern);
-       len += sprintf(buf + len, "Hop index         = %d \n", pfh->hop_index);
+       seq_printf(m, "Hop set           = %d \n", pfh->hop_set);
+       seq_printf(m, "Hop pattern       = %d \n", pfh->hop_pattern);
+       seq_printf(m, "Hop index         = %d \n", pfh->hop_index);
        p += p[1] + 2;
    }
    else {
-       len += sprintf(buf + len, "Parse beacon failed at FH param element\n");
-       return len;
+       seq_puts(m, "Parse beacon failed at FH param element\n");
+       return 0;
    }
    } else {
-       len += sprintf(buf + len, "No beacons received\n");
+       seq_puts(m, "No beacons received\n");
    }
-   return len;
+   return 0;
}

+static int ray_cs_proc_open(struct inode *inode, struct file *file)
+{
+ return single_open(file, ray_cs_proc_show, NULL);
+}

```

```

+
+static const struct file_operations ray_cs_proc_fops = {
+ .owner = THIS_MODULE,
+ .open = ray_cs_proc_open,
+ .read = seq_read,
+ .llseek = seq_lseek,
+ .release = single_release,
+};
#endif

/*=====
==*/
static int build_auth_frame(ray_dev_t *local, UCHAR *dest, int auth_type)
@@ -2815,7 +2826,7 @@ static int __init init_ray_cs(void)
#ifdef CONFIG_PROC_FS
    proc_mkdir("driver/ray_cs", NULL);

-   create_proc_info_entry("driver/ray_cs/ray_cs", 0, NULL, &ray_cs_proc_read);
+   proc_create("driver/ray_cs/ray_cs", 0, NULL, &ray_cs_proc_fops);
    raycs_write("driver/ray_cs/ssid", write_essid, NULL);
    raycs_write("driver/ray_cs/net_type", write_int, &net_type);
    raycs_write("driver/ray_cs/translate", write_int, &translate);
--
1.5.4.3

```
